RDF/RDFS Tutorial

Introduction

The *Resource Description Framework* (RDF) is recommended by the World *Wide Web Consortium* (W3C) to model meta-data about the resources of the web. It is described in both documents [1] and [2].

[1] Graham Klyne, Jeremy J. Carroll (Eds.). Resource Description Framework (RDF): Concepts and Abstract Syntax.

http://www.w3.org/TR/rdf-concepts/

[2] Dan Brickley, R. V. Guha (Eds.). RDF Vocabulary Description Language 1.0: RDF Schema.

http://www.w3.org/TR/rdf-schema/

The former focuses on syntactical aspects while the latter addresses the definition of vocabularies (often named *schemas*).

Here we use a slightly different plan directed to the goals of the KB course (using RDF in knowledge representation systems).

1. Model

1.1. URIs

RDF identifies resources with standard *Uniform Resource Identifiers* (URI), but RDF uses what we will call *qualified* URIs, that is, URIs with an optional fragment identifier (a text added to the URI with a “#” between them).

The fragment identifier returns “a property of the data resulting from a retrieval action”; however, RDF considers every qualified URI (with or without fragment identifier) as a full resource by itself.

1.2. Triples and graph

The base element of the RDF model is the triple: a resource (the *subject*) is linked to another resource (the *object*) through an arc labeled with a third resource (the *predicate*). We will say that <subject> has a *property* <predicate> *valued* by <object>.

For example, the triple in figure 1 could be read as “Champin is the creator of index.html”.

Figure 1: A triple

All the triples result in a directed graph, whose nodes and arcs are all labeled with qualified URIs. Note in figure 2 that a resource may have more than one value for a given property.

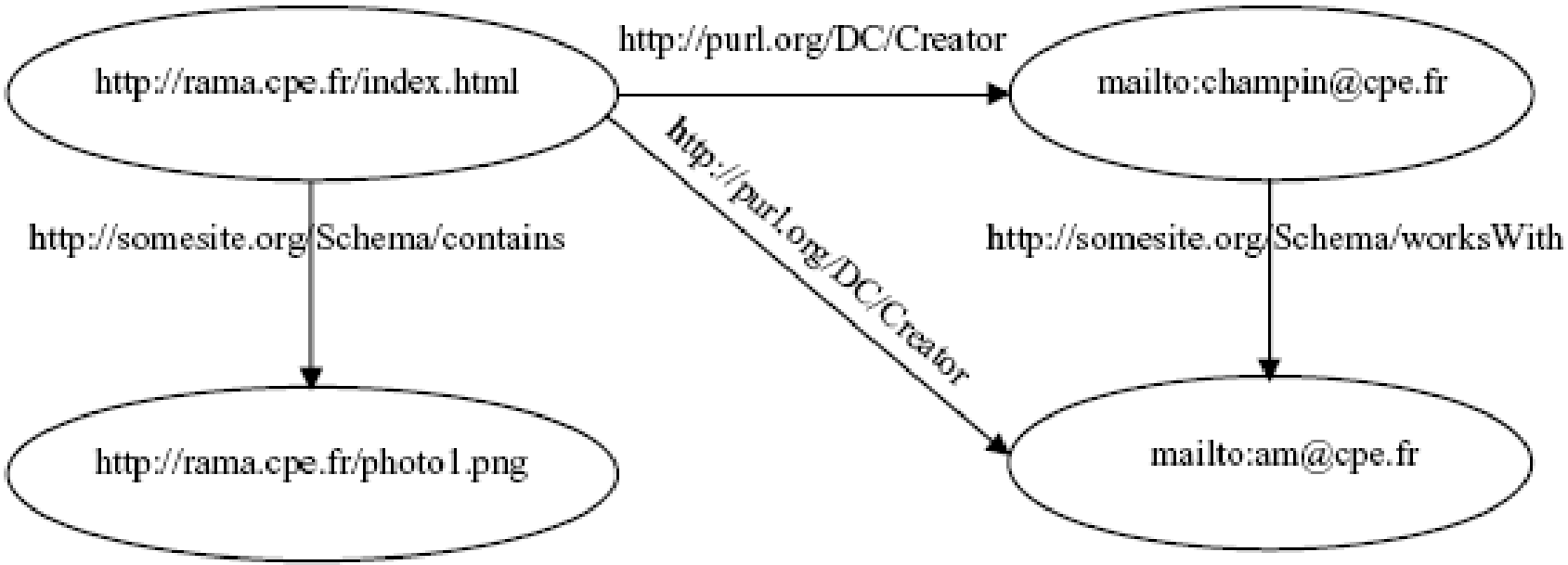

Figure 2: An RDF graph

1.3. Literals

In the RDF recommendation, targets of the graph can be pieces of text instead of resources; those pieces of text are called *literals*.

2. Concepts and vocabulary

We can distinguish three kinds of concepts in RDF: ***fundamental concepts***, ***schema-definition concepts*** (useful for defining new vocabularies) and ***utility concepts*** (concepts which are not absolutely necessary, but likely to be useful in any application domain).

All these concepts have been given a URI. ***These URIs are defined as fragment identifiers of the URIs of the W3C documents defining RDF.***

For the sake of clarity, we will rather use the XML non-expanded notation.

That is, prefixes rdf: and rdfs: will be used instead of

http://www.w3.org/TR/1999/REC-rdf-syntax-19990222#

and

http://www.w3.org/TR/1999/PR-rdf-schema-19990303#

respectively.

The membership of one or another namespace may not always seem logical, and must have historical reasons mostly.

2.1. Fundamental concepts

2.1.1. rdf:Resource

RDF is about describing resources; according to [1], “resources are always named by URIs” and “anything can have a URI”. So RDF can theoretically be used to describe anything. Yet it was mainly designed to handle “network retrievable” resources.

Some authors underline that “the resource is the conceptual mapping to an entity (…), not necessarily the entity which corresponds to that mapping *at any particular instance in time*”. However most of the time we are interested in entities themselves. It is therefore important to note that the meta-data we express about resources may require different levels of interpretation, which may be valid in a certain context only.

For example, the URI

`http://www.w3.org/Icons/WWW/w3c_main`

returns the W3C logo in the PNG or GIF format, depending on the browser being used.

Another example is the daily weather report, whose URL would return a different page each day.

It follows that the interpretation of resources (and therefore of RDF triples) is highly contextual. We can define the notion of *stable resource* as follows: stability for a resource is the property of being the same in any context, from the point of view of a user (or a community of users). This definition is still very contextual: it is dependant on the users we are considering, more precisely on the task they have to accomplish.

For example, from the point of view of a standard reader, the W3C logo is stable, since the GIF and PNG versions look the same, but the weather report is not stable. On the other hand, someone interested only in image formats may consider the W3C logo unstable and the weather report stable – assuming the weather report is always generating images in the same format.

2.1.2. rdf:Property

The properties are resources used as predicate of triples; the semantics of a triple clearly depends on the property used as predicate. Two things are very important with the concept of property.

First, RDF considers properties as first class object, unlike object modeling languages, where properties are attributes of a class. Even though the concept of class exists in RDF (see subsection 2.2), properties can be defined and used independently of classes.

Secondly, the fact that properties are resources allows to describe them with RDF itself. This will be widely used by the following concepts.

2.1.3. rdf:Statement

A statement is a resource reifying a triple. Such a resource must have at least 3 properties: **rdf:subject**, **rdf:object** and **rdf:predicate**, valued by the corresponding resources.

The reification of triples may seem a utility concept rather than a fundamental concept. Nevertheless it is defined as a part of the model in the W3C recommendation. This supports the will to use RDF as its own meta-system, to make every element of RDF describable in RDF itself.

2.2. Schema definition concepts

All these concepts are defined in [2], the second document of the W3C, to allow the definition of schemas, that is, vocabularies of resources to use with RDF. Not all agents will need to be aware of these concepts: specialized agents, limited to using a predefined vocabulary, will not.

In schemas, new resources can be defined as *specialization* of old ones, thus allowing to infer implicit triples. Schemas also constrain the context in which defined resources may be used, inducing the notion of *schema validity*. We will see that these two notions can be seen as one, in a point of view based on first-order logic.

They all can be expressed as rules allowing to infer new facts (basically, new triples or *negations* of triples). In these rules, the 3-ary logical predicate

T(*subject*, *predicate*, *object*)

will be used to represent a believed triple.

2.2.1. rdfs:subPropertyOf

Any property denotes a relation between resources (the set of resource couples linked by an arc labeled with the property).

rdfs:subPropertyOf applies to properties and must be interpreted as the subset relation between the relations they denote.

Thus the following rule stands:

$\forall s,p_1,o,p_2 \quad T(s,p_1,o) \wedge T(p_1,\text{rdfs:subPropertyOf},p_2) \Rightarrow T(s,p_2,o)$

For example, if “mother” is a sub-property of “parent”, any triple having “mother” as predicate must also be considered as having “parent” as predicate.

This property is very important in schema definitions for interoperability between RDF agents.

In the example above, an agent not knowing the semantics of “mother” could at least treat it as “parent” (assuming it knows the semantics of “parent”).

Since rdfs:subPropertyOf denotes a subset relation, the transitivity rule also stands:

$\forall p_1,p_2,p_3$

$\mathrm{T}(p_1,\text{rdfs:subPropertyOf},p_2) \wedge$

$\mathrm{T}(p_2,\text{rdfs:subPropertyOf},p_3) \Rightarrow$

$\mathrm{T}(p_1,\text{rdfs:subPropertyOf},p_3)$

Note that it is considered invalid by [2] to have cycles in rdfs:subPropertyOf, though it doesn't define a way to express this constraint in RDF. Anyway, the corresponding logical rule is the following (since any cycle would result, with transitivity, in a property being its own sub-property):

$$\forall p \quad \neg \mathrm{T}(p, \text{rdfs:subPropertyOf}, p)$$

Note also that there is no standard URI for the universal property (superproperty of any property).

2.2.2. rdfs:Class, rdf:type and rdfs:subClassOf

Classes are resources denoting a set of resources, by the mean of the property rdf:type (instances have property rdf:type valued by the class). Since all sets of resources presented in this section are resources (they have a URI), they have by definition the property rdf:type valued by rdfs:Class.

Classes are structured the same way as properties, in a subset hierarchy denoted by the property rdfs:subClassOf. As for rdfs:subPropertyOf, cycles must not exist though it could be used to express equivalence, but contrary to the property hierarchy, the class hierarchy has a maximum element: it is rdf:Resource (so any class implicitly has rdfs:subClassOf valued by rdf:Resource).

The following rules, similar to the rules related to rdfs:subPropertyOf, stand:

$\forall i, c_1, c_2 \quad \mathrm{T}(i, \text{rdf:type}, c_1) \wedge \mathrm{T}(c_1, \text{rdfs:subClassOf}, c_2) \Rightarrow \mathrm{T}(i, \text{rdf:type}, c_2)$

$\forall c_1, c_2, c_3 \quad \mathrm{T}(c_1, \text{rdfs:subClassOf}, c_2) \wedge \mathrm{T}(c_2, \text{rdfs:subClassOf}, c_3) \Rightarrow \mathrm{T}(c_1, \text{rdfs:subClassOf}, c_3)$

$\forall c \quad \neg \mathrm{T}(c, \text{rdfs:subClassOf}, c)$

2.2.3. rdfs:domain and rdfs:range

These properties apply to properties and must be valued by classes. They are used to restrict the set of resources that may have a given property (the property's *domain*) and the set of valid values for a property (its *range*).

A property may have as many values for rdfs:domain as needed, but no more than one value for rdfs:range:

$\forall p, r_1, r_2 \quad \mathrm{T}(p, \text{rdfs:range}, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg \mathrm{T}(p, \text{rdfs:range}, r_2)$

For a triple to be valid, the object must match the range (if any) of the predicate (that is, it must have rdf:type valued by the corresponding class or one of its subclasses), and the subject must match at least one of the domains (if any) of the predicate.

Note that if the predicate has super-properties, this must also be checked recursively for all of them. This can be logically expressed by:

$\forall s,p,o \quad \mathrm{T}(s,p,o) \wedge \exists d\ \mathrm{T}(p,\text{rdfs:domain},d) \Rightarrow$

$\exists d'\ (\mathrm{T}(p,\text{rdfs:domain},d') \wedge \mathrm{T}(s,\text{rdf:type},d'))$

$\forall s,p,o,r \quad \mathrm{T}(s,p,o) \wedge \mathrm{T}(p,\text{rdfs:range},r) \Rightarrow$

$\mathrm{T}(o,\text{rdf:type},r)$

2.2.4. rdfs:Literal

[2] defines a resource rdfs:Literal, denoting the set of literals, declared as a class (though literals are not resources, according to the recommendation). Its intended use is to be the range of properties.

2.3. Utility concepts

These concepts may have been defined in external schemas, but since they are of very common use, they have been defined once for all in the core schema.

2.3.1. rdfs:Container

Containers are collections of resources. They are modeled by an instance of one of the three subclasses of rdfs:Container: rdf:Bag (an unordered collection), rdf:Seq (an ordered collection) or rdf:Alt (an alternative). Membership is modeled by automatically generated properties rdf:_1, rdf:_2, etc. These properties are all instances of rdfs:ContainerMembershipProperty, a subclass of rdf:Property.

2.3.2. rdfs:ConstraintResource and rdfs:ConstraintProperty

It can be interesting for an RDF agent to be informed that an unknown resource (or more specifically a property) is defining a validity constraint. The set of such resources is rdfs:ConstraintResource. Its subclass rdfs:ConstraintProperty is of course a subclass of rdf:Property too. Properties rdfs:domain and rdfs:range defined above are instances of rdfs:ConstraintProperty.

2.3.3. rdfs:seeAlso and rdfs:isDefinedBy

A given resource may be described in more than one place over the internet. The rdfs:seeAlso property can be used to point to alternative descriptions of the subject resource. Its sub-property rdfs:isDefinedBy more specifically points to an original or authoritative description.

2.3.4. rdfs:label and rdfs:comment

It can be useful to describe a resource with human readable text in addition to “pure” RDF properties; this is the role of rdfs:label and rdfs:comment. The former is used to give a human-readable name of a resource, the latter - to give a longer description. Note that they may have multiple values for internationalization needs.

3. XML syntax

This section describes the XML syntax recommended by [1]. It uses XML namespace notations, but expanded names are obtained simply by concatenating the namespace to the element name. Hence we will use the same convention as in the previous section for prefixes rdf: and rdfs:.

An RDF document is a list of descriptions. Each description applies usually to one resource, and contains a list of properties. Property values are either URIs, literals or other Descriptions.

In XML, RDF meta-data are embedded in an element named rdf:RDF. This element contains a sequence of elements named rdf:Description. These elements can have one of the two attributes:rdf:about or rdf:ID (but not both).

- rdf:about is used to describe any resource; its value is either an absolute or a relative URI.

```
<rdf:Description about="http://rama.cpe.fr/index.html">
   ...
</rdf:Description>
```

- rdf:ID is used to define a resource; its value is a fragment identifier (without the “#” character) to be added to the XML document URI. A resource may not be defined more than once.

```
<rdf:Description ID="foo">
  ...
</rdf:Description>
```

- a description without rdf:about nor rdf:ID is said to describe an *anonymous* resource.

```
<rdf:Description>
   ...
</rdf:Description>
```

An element rdf:Description contains a sequence of XML elements. These elements are interpreted as properties, whose predicate's URI is the expanded name of the element. If the element is empty, it must have an attribute rdf:resource whose value is the object's URI (see 1st dc:Creator in fig. 3). Else, it can contain plain text (then interpreted as a literal – see dc:Title in fig. 3) or a single embedded rdf:Description element (see 2nd dc:Creator in fig. 3).

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/DC/"
         xmlns:os="http://somesite.org/Schema/">
  <rdf:Description about="http://rama.cpe.fr/index.html">
    <dc:Creator rdf:resource="mailto:am@cpe.fr"/>
    <dc:Title> Index of my web site </dc:Title>
    <dc:Creator>
      <rdf:Description about="mailto:champin@cpe.fr">
        <os:worksWith rdf:resource="mailto:am@cpe.fr"/>
      </rdf:Description>
    </dc:Creator>
  </rdf:Description>
</rdf:RDF>
```

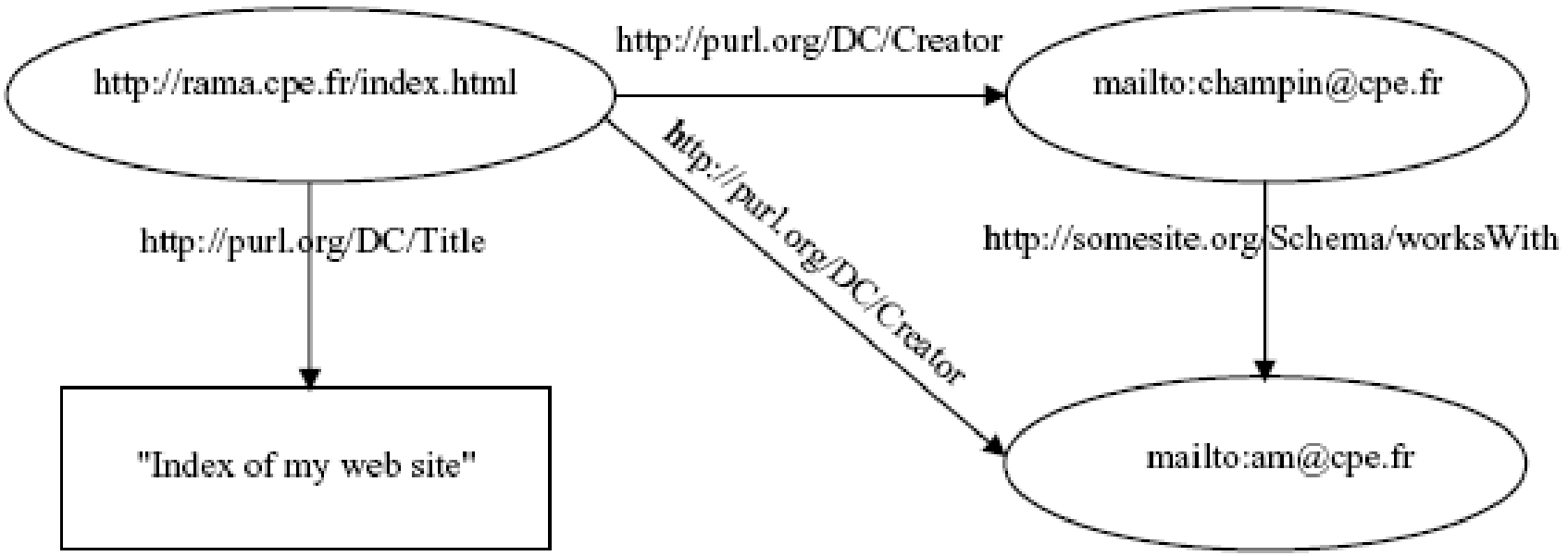

Figure 3: XML syntax simple example